Homework 1

Asymptotics and Number Theoretic Algorithms

Deadline: February 18, 11:59pm.
Available points: 120. Perfect score: 100.

Homework Instructions:

Teams: Homeworks should be completed by teams of students - three at most. No additional credit will be given for students that complete a homework individually. Please inform Athanasios Krontiris about the members of your team as soon as possible (email: ak979 AT cs.rutgers.edu).

Submission Rules: Submit your solutions electronically as a PDF document through `sakai.rutgers.edu`. Do not submit Word documents, raw text, or hardcopies etc. Make sure to generate and submit a PDF instead. Each team of students should submit only a single copy of their solutions and indicate all team members on their submission. Failure to follow these rules will result in lower grade in the assignment.

Late Submissions: No late submission is allowed. If you don't submit a homework on time, you get 0 points for that homework.

Extra Credit for LaTeX: You will receive 10% extra credit points if you submit your answers as a typeset PDF (using LaTeX, in which case you should also submit electronically your source code). Resources on how to use LaTeX are available on the course's website. There will be a 5% bonus for electronically prepared answers (e.g., on MS Word, etc.) that are not typeset.

25% Rule: For any homework problem (same will hold for exam questions), you can either attempt to answer the question, in which case you will receive between 0 and 100% credit for that question, or you can write "I don't know", in which case you receive 25% credit for that question. Leaving the question blank is the same as writing "I don't know." You can and will get less than 25% credit for a question that you answer erroneously.

Handwritten Reports: If you want to submit a handwritten report, scan it and submit a PDF via Sakai. We will not accept hardcopies. If you choose to submit handwritten answers and we are not able to read them, you will not be awarded any points for the part of the solution that is unreadable.

Precision: Try to be precise. Have in mind that you are trying to convince a very skeptical reader (and computer scientists are the worst kind...) that your answers are correct.

Collusion, Plagiarism, etc.: Each team of students must prepare its solutions independently from other teams, i.e., without using common notes or worksheets with other students or trying to solve problems in collaboration with other teams. You must indicate any external sources you have used in the preparation of your solution. Do not plagiarize online sources and in general make sure you do not violate any of the academic standards of the course, the department or of the university (the standards are available through the course's website). Failure to follow these rules may result in failure in the course.

Part A (20 points)

Problem 1: In each of the following situations indicate whether $f = O(g)$ or $f = \Omega(g)$ or $f = \Theta(g)$:

1. $f(n) = \sqrt{2^{7n}},\ g(n) = \lg(7^{2n})$
2. $f(n) = 2^{nln(n)},\ g(n) = n!$
3. $f(n) = lg(lg^* n),\ g(n) = lg^*(lgn)$
4. $f(n) = \frac{lgn^2}{n},\ g(n) = lg^* n$
5. $f(n) = 2^n,\ g(n) = n^{lgn}$
6. $f(n) = 2^{\sqrt{lgn}},\ g(n) = n(lgn)^3$
7. $f(n) = e^{cos(n)},\ g(n) = lgn$
8. $f(n) = lgn^2,\ g(n) = (lgn)^2$
9. $f(n) = \sqrt{4n^2 - 12n + 9},\ g(n) = n^{\frac{3}{2}}$
10. $f(n) = \sum_{k=1}^{n} k, g(n) = (n+2)^2$

Note: The $lg^* n$ function is the number of times the logarithm function must be iteratively applied before the result is less than or equal to 1, i.e., $lg^* n = 0$ if $n \leq 1$ and $lg^* n = 1 + lg^{(}lgn)$ if $n > 1$. For instance, $lg^* 4 = 2$, $lg^* 16 = 3$, $lg^* 65536 = 4$, etc.

Note: To check the relative asymptotic behavior between two functions $f(n)$ and $g(n)$, we may use transformation by a monotonically increasing function. For instance, we usually use logarithmic functions for simplifying the expressions. This is because logarithmic functions convert multiplication to addition and power to multiplication. Nevertheless, you can apply any monotonically increasing function in the range from 0 to infinity to simplify the expressions. If after the application of the transformation, one expression dominates the other as n goes to infinity, then this is also true for the original expressions. Examples include a square function x^2 or even an exponential function e^x. For instance, to compare $f(n)$ and $g(n)$, it is sufficient to compare $f^2(n)$ and $g^2(n)$.

Algorithm 1: `Number_Theoretic_Algorithm` (integer n)

```
$N \leftarrow Random\_Sample(0, 2^n - 1)$;
if $N$ is even then
    $N \leftarrow N + 1$;
$m \leftarrow N \mod n$;
for $j \leftarrow 0$ to $m$ do
    if Greatest_Common_Divisor($j, N$) $\neq 1$ then
        return FALSE;
    Compute $x, z$ so that $N - 1 = 2^z \cdot x$ and $x$ is odd;
    $y_0 \leftarrow (N - 1 - j)^x \mod N$;
    for $i \leftarrow 1$ to $m$ do
        $y_i \leftarrow y_{i-1}^2 \mod N$;
        $y_i \leftarrow y_i + y_{i-1} \mod N$;
    if Low_Error_Primality_Test($y_m$) == FALSE then
        return FALSE;
return TRUE;
```

Problem 2: Compute the asymptotic running time of the above algorithm as a function of its input parameter, given:

- The running times of integer arithmetic operations (e.g., multiplication of two large n-bit numbers is $O(n^2)$).
- Assume that sampling a number N is an operation linear to the number of bits needed to represent this number.

Do not just present the final result. For each line of pseudo-code indicate the best running time for the corresponding operation given current knowledge from lectures and recitations and then show how the overall running time emerges.

Part B (30 points)

Problem 3:

- Consider that we have a tree data structure T_m^N, where every node can have at most m children and the tree has at most N nodes total. Compute a lower bound for the height of the tree.

- Consider two such trees T_m^N and $T_{m'}^N$ that are "perfect", i.e., every node has exactly m and m' children correspondingly. Now, consider the functions $h_m(N)$ and $h_{m'}(N)$ that express the heights of these perfect trees for different values of N. What is the asymptotic behavior of h_m relative to $h_{m'}$ and under what conditions?

- Consider the following rule for modular exponentiation modulo a number N, where x is in the order of 2^m, y is in the order of 2^n and N is in the order of 2^o. What is the running time of computing the result according to this rule?

$$x^y \mod N \equiv \begin{cases} (x^{\lfloor \frac{y}{2} \rfloor})^2 \mod N, & \text{if y is even} \\ x \cdot (x^{\lfloor \frac{y}{2} \rfloor})^2 \mod N, & \text{if y is odd} \end{cases}$$

Problem 4:

- Compute the following: $2^{902} \bmod 7$.

- Find the modulo multiplicative inverse of 11 mod 120, 13 mod 45, 35 mod 77, 9 mod 11, 11 mod 1111.

- Assume that for a number x the following property is true: $\forall\ y \in [1, x-1] :\ gcd(x,y) = 1$. Compute the running time of an efficient algorithm for finding all the inverses modulo x^m from the set $\{0, 1, \ldots, x^m - 1\}$ that exist.

Problem 5:

- Assume two positive integers x < y. Then the pairs $(5x+3y, 3x+2y)$ and (x, y) have the same greater common divisor. True or False, explain.

- Consider the following sequence of numbers: $s_n = 1 + \Pi_{i=0}^{n-1} s_i$, where $s_0 = 2$. Prove that any two numbers in this sequence are relatively prime.

Part C (40 points)

Problem 6: Our goal is to assign $(2^{31} - 1)$ integers into 256 slots $\{0, 1, \ldots, 255\}$ and achieving the properties of universal hashing. Notice that 256 is not a prime number. Assume the following approach towards this objective.

Consider a 32-bit integer y and the following set $\mathcal{M}$ of hash functions, so that:

- each $m \in \mathcal{M}$ corresponds to a unique 8×32 matrix M having elements only 0 or 1.

- and $m(y) = M \cdot y \bmod 2$, i.e., multiply the matrix with the 32-bit vector corresponding to number y and then apply the modulo operation on each bit of the resulting vector.

Such functions map a 32-bit vector into an 8-bit vector, which can then be interpreted as an 8-bit number that indicates the original number's slot in the range $0 \sim 255$. Notice that there are $2^{8\times32} = 2^{256}$ different $\{0,1\}$ matrices in the set $\mathcal{H}$.

Provide the following:

- A proof that the hash function family $\mathcal{M}$ is universal. [Hint: You have to show that the hash function family has the "consistent" and "random" property. The first is rather obvious to show. The second requires that before fixing the matrix M the probability of two distinct vectors x and y to be mapped to the same index $m(x) = m(y)$ is equal to the probability of them assigned to the same index, when indices $m(x)$ and $m(y)$ are sampled uniformly at random. So you need to consider how two different vectors x and y can be mapped to the same index $m(x)$ and $m(y)$ when $m(x) \equiv Mx \mod 2$ and $m(y) \equiv My \mod 2$.]
- A comparison to the universal hash function family described in DPV chapter 1.5.2. How many random bits are needed here?

Problem 7: Answer the following sequence of problems:

- Assume that number n is prime, then all numbers $1 \leq x < n$ are invertible modulo n. Which of these numbers are their own inverse modulo n? [Hint: x is its own inverse modulo n if $x^2 \equiv 1 \bmod n$ which means $x^2 - 1 \equiv 0 \mod n$. Note that when a number k can be written as the product $k = k_1 \cdot k_2$, then the factorization of k corresponds to the product of the factorizations of k_1 and k_2. The factorization of a number k corresponds to the product of primes that result in k. All these primes must be smaller then k.]

- Show that $(n-1)! \equiv -1 \mod n$ for prime n. [Hint: Consider all the pairs of numbers that are smaller than n, which are multiplicative inverses modulo n, multiply these numbers and compute the result of this product modulo n. You have to show in this process that the multiplicative inverse of a number x exists and is unique. Try then to generate the $(n-1)!$ on the left side and see what you get on the right side.]

- Show that if n is not prime, then $(n-1)! \not\equiv -1 \mod n$. [Hint: What does it mean that n is not prime in terms of the numbers $1 \leq x < n$? If you try to compute $(n-1)!$ the same way as in the above question given that n is not prime, what will happen?]

- The above process can be used as a primality test instead of Fermat's Little theorem as it is an if-and-only-if condition for primality. Why can't we immediately base a primality test on this rule? [Tip: Even if you are not able to answer the previous two questions, you should still be able to argue about this question.]

Part D (30 points)

Problem 8:

A. Make a table with three columns. The first column is all numbers from 0 to 36. The second is the residues of these numbers modulo 5; the third column is the residues modulo 7.

B. Consider two different prime numbers x and y. Show that the following is true: for every pair of numbers m and n so that: $0 \leq m < x$ and $0 \leq n < y$, there is a unique integer q, where $0 \leq q < xy$, so that:

$$q \equiv m \pmod{x}$$
$$q \equiv n \pmod{y}$$

[Hint: Think how many q's in the range $[0, xy]$ can have the same result modulo x and modulo y and count how many q's there are.]

C. The previous problem asks to go from q to (m, n). It is also possible to go the other way. In particular, show the following:

$$q = (m \cdot y \cdot (y^{-1} \bmod x) + n \cdot x \cdot (x^{-1} \bmod y)) \bmod xy$$

[Hint: Ensure that if the above is true then the expressions in section B are also true. Consider the values of the following terms: $c_x = y \cdot (y^{-1} \bmod x)$ and $c_y = x \cdot (x^{-1} \bmod y)$ and their values $\bmod x$ and $\bmod y$.]

D. What happens in the case of three primes x, y and z? Do the above properties still hold? If they do, how do they look like in this case?

Problem 9: There is an office, in which every member uses RSA to conduct secure communication with others. For example, when Alice wants to send Bob a message, she will first use Bob's public key to encrypt the message, then sends it to Bob, Bob then uses his private key to decrypt the message. In order to make things easy, the office maintains a directory listing every member's public key, everybody in the office has access to it. A public key looks like (N, e) (as defined in DPV-1.4.2).

One day, Alice sent a message (smaller than any N) to Bob, Charlie, and David via the RSA based secure communication, but the encrypted messages were intercepted by the webmaster Mallory. Mallory used some network tricks and found out who the receivers were, then he checked their public keys in the directory. This is what Mallory got:

Receiver	Encrypted message	Public key
Bob	674	(3337, 3)
Charlie	36	(187, 3)
David	948	(1219, 3)

Finally, Mallory recovered the original message. How did Mallory do this? What is the original message?

Note: Please do not directly use factorization. It is possible to do so because the numbers are small. You can use factorization to validate your result.

[Hint: Consider how the three transmitted messages are computed. The result of the previous problem 8 is useful here. Be careful with the details and under which conditions you apply certain expressions.]